

# A combinatorial algorithm for all-pairs shortest paths in directed vertex-weighted graphs with applications to disc graphs

Andrzej Lingas<sup>1</sup> and Dzmitry Sledneu<sup>2</sup>

<sup>1</sup> Department of Computer Science, Lund University, 22100 Lund.  
 Andrzej.Lingas@cs.lth.se. Fax +46 46 13 10 21

<sup>2</sup> The Centre for Mathematical Sciences, Lund University, 22100 Lund, Sweden.  
 Dzmitry.Sledneu@math.lu.se

**Abstract.** We consider the problem of computing all-pairs shortest paths in a directed graph with real weights assigned to vertices.

For an  $n \times n$  0 – 1 matrix  $C$ , let  $K_C$  be the complete weighted graph on the rows of  $C$  where the weight of an edge between two rows is equal to their Hamming distance. Let  $MWT(C)$  be the weight of a minimum weight spanning tree of  $K_C$ .

We show that the all-pairs shortest path problem for a directed graph  $G$  on  $n$  vertices with nonnegative real weights and adjacency matrix  $A_G$  can be solved by a combinatorial randomized algorithm in time<sup>3</sup>

$$\tilde{O}(n^2 \sqrt{n + \min\{MWT(A_G), MWT(A_G^t)\}})$$

As a corollary, we conclude that the transitive closure of a directed graph  $G$  can be computed by a combinatorial randomized algorithm in the aforementioned time.

We also conclude that the all-pairs shortest path problem for uniform disk graphs, with nonnegative real vertex weights, induced by point sets of bounded density within a unit square can be solved in time  $\tilde{O}(n^{2.75})$ .

## 1 Introduction

The problems of finding shortest paths and determining their lengths are fundamental in algorithms. They have been extensively studied in algorithmic graph theory. A central open question in this area is if there is a substantially subcubic in the number of vertices algorithm for the all-pairs shortest path problem for directed graphs with real edge weights (APSP) in the addition-comparison model [24,27]. For several special cases of weights and/or graphs substantially subcubic algorithms for the APSP problem are known [3,8,23,25,26,27]. However, in the general case the fastest known algorithm due to Chan [8] (see also

<sup>3</sup> The notation  $\tilde{O}(\cdot)$  suppresses polylogarithmic factors and  $B^t$  stands for the transposed matrix  $B$ .

[9]) runs in time  $O(n^3 \log^3 \log n / \log^2 n)$ , achieving solely a moderate polylogarithmic improvement over the  $O(n^3)$  bound yielded by Floyd-Warshall and Johnson's algorithms [1,27].

The situation is different for directed graphs with real vertex weights. Recently, Chan has shown that the APSP problem for the aforementioned graphs can be solved in time  $O(n^{2.844})$  [8] and Yuster has slightly improved the latter bound to  $O(n^{2.842})$  by using an improved bound on rectangular multiplication [25].

The basic tool in achieving substantially subcubic upper bounds on the running time for the APSP for directed graphs with constrained edge weights or real vertex weights are the fast algorithms for arithmetic square and rectangular matrix multiplication [10,14]. One typically exploits here the close relationship between the APSP problem and the so called distance or  $(\min, +)$  product [3,23,24,25,27,26].

Unfortunately, these fast algorithms for matrix multiplication, yielding equally fast algorithms for Boolean matrix product, are based on recursive algebraic approaches over a ring difficult to implement. Thus, another central question in this area is whether or not there is a substantially subcubic *combinatorial* (i.e., not relying on ring algebra) algorithm for the Boolean product of two  $n \times n$  Boolean matrices [4,22,24]. Again, the fastest known combinatorial algorithm for Boolean matrix product due to Bansal and Williams [4] running in time  $O(n^3 \log^2 \log n / \log^{9/4} n)$  achieves solely a moderate polylogarithmic improvement over the trivial  $O(n^3)$  bound. On the other hand, several special cases of Boolean matrix product admit substantially subcubic combinatorial algorithms [5,13,20].

In particular, Björklund et al. [5] provided a combinatorial randomized algorithm for Boolean matrix product which is substantially subcubic in case the rows of the first  $n \times n$  matrix or the columns of the second one are highly clustered, i.e., their minimum spanning tree in the Hamming metric has low cost. More exactly, their algorithm runs in time  $\tilde{O}(n(n+c))$ , where  $c$  is the minimum of the costs of the minimum spanning trees for the rows and the columns, respectively, in the Hamming metric. It relies on the fast Monte Carlo methods for computing an approximate minimum spanning tree in the  $L_1$  and  $L_2$  metrics given in [16,17].

The assumption that the input directed graph is highly clustered in the sense that the minimum spanning tree of the rows or columns of its adjacency matrix in the Hamming metric has a subquadratic cost does not yield any direct applications of the algorithm of Björklund et al. [5] to shortest path problems, not even to the transitive closure. The reason is that the cost of the analogous minimum spanning tree can grow dramatically in the power graphs<sup>4</sup> of the input graph. In particular, we cannot obtain directly an upper time-bound on the transitive closure of Boolean matrix corresponding to that for the Boolean matrix product from [5] by applying the asymptotic equality between the time

---

<sup>4</sup> In the  $i$ -th power graph there is an edge from  $v$  to  $u$  if there is a path composed of at most  $i$  edges from  $v$  to  $u$  in the input graph.

complexity of matrix product over a closed semi-ring and that of its transitive closure over the semi-ring due to Munro [21]. The reason is the dependence of the upper bound from [5] on the cost of the minimum spanning tree.

In this paper, we extend the idea of the method from [5] to include a mixed product of a real matrix with a Boolean one. We combine the aforementioned extension with the ideas used in the design of subcubic algorithms for important variants of the APSP problem [3,26], in particular those for directed graphs with vertex weights [8,25], to obtain not only a substantially subcubic combinatorial algorithm for the transitive closure but also for the APSP problem in highly clustered directed graphs with real vertex weights.

For an  $n \times n$  0 – 1 matrix  $C$ , let  $K_C$  be the complete weighted graph on the rows of  $C$  where the weight of an edge between two rows is equal to their Hamming distance. Let  $MWT(C)$  be the weight of a minimum weight spanning tree of  $K_C$ . We show that the all-pairs shortest path problem for a directed graph  $G$  on  $n$  vertices with nonnegative real weights and an adjacency matrix  $A_G$  can be solved by a combinatorial randomized algorithm in  $\tilde{O}(n^2 \sqrt{n} + \min\{MWT(A_G), MWT(A_G^t)\})$  time. It follows in particular that the transitive closure of a directed graph  $G$  can be computed by a combinatorial randomized algorithm in the aforementioned time.

Our algorithms are of Monte Carlo type and by increasing the polylogarithmic factor at the time bounds, the probability that they return a correct output within the bounds can be amplified to  $1 - \frac{1}{n^\alpha}$ , where  $\alpha \geq 1$ .

*Since there are no practical or combinatorial substantially subcubic-time algorithms not only for the APSP problem but even for the transitive closure problem for arbitrary directed graphs at present, our simple adaptive method might be a potentially interesting alternative for a number of graph classes.*

As an example of an application of our method, we consider the APSP problem for uniform disk graphs, with nonnegative real vertex weights, induced by point sets of bounded density within a unit square. We obtain a combinatorial algorithm for this problem running in time  $O(\sqrt{r}n^{2.75})$ , where  $r$  is the radius of the disks around the vertices in a unit square.

The recent interest in disk graphs, in particular uniform disk graphs, stems from their applications in wireless networks. In this context, the restriction to point sets of bounded density is quite natural. In [11], Fürer and Kasiviswanathan provided a roughly  $O(n^{2.5})$ -time preprocessing for *approximate*  $O(\sqrt{n})$ -time distance queries in arbitrary disk graphs.

Our paper is structured as follows. In the next section, we show a reduction of the APSP problem for directed graphs with real vertex-weights to a mixed matrix product of a distance matrix over reals with the 0 – 1 adjacency matrix. In Section 3, we present an algorithm for such a mixed product which generalizes that for the Boolean matrix product from [5] and runs in subcubic time if the input 0 – 1 matrix is highly clustered. By combining the results of Sections 2,3, we can derive our main results in Section 4. In the next section, we present the application of our method to uniform disk graphs induced by point sets of bounded density. We conclude with final remarks.

## 2 A Reduction of APSP to Mixed Matrix Products

### 2.1 The APSP problem

Formally, the All-Pairs Shortest Paths problem (APSP) in a directed graph  $G = (V, E)$  with real weights  $w(v)$  associated to vertices  $v \in V$  is to compute the  $|V| \times |V|$  distance matrix  $D_G$  such that  $D_G(v, u)$  is the distance  $\delta_G(v, u)$  from  $v$  to  $u$  in  $G$ , i.e., the minimum total weight of vertices on a path from  $v$  to  $u$  in  $G$ . An additional goal of the APSP problem is to compute a concise data structure representing the shortest paths.

Note that  $\delta_G(v, u)$  is equal to the minimum total weight of inner vertices on a path from  $v$  to  $u$  in  $G$  increased by the weights of  $v$  and  $u$ .

We shall assume  $|V| = n$  throughout the paper.

For  $i = 0, 1, \dots, n-1$ , let  $\delta_G^i(v, u)$  be the distance from  $v$  to  $u$  on paths consisting of at most  $i$  edges, i.e., the minimum total weight of vertices on a path from  $v$  to  $u$  having at most  $i$  edges in  $G$ . Next, let  $D_G^i$  be the  $|V| \times |V|$  matrix such that  $D_G^i[v, u]$  is equal to  $\delta_G^i(v, u)$ .

For convention, we assume  $\delta_G^0(v, v) = 0$  and  $\delta_G^0(v, u) = +\infty$  for  $v \neq u$ . Hence,  $D_G^0$  has zeros on the diagonal and  $+\infty$  otherwise. In  $D_G^1$ , all the entries  $D_G^1[v, u]$  where  $(v, u) \in E$  are set to  $w(v) + w(u)$  instead of  $+\infty$ . Thus, both  $D_G^0$  and  $D_G^1$  can be easily computed in time  $O(n^2)$ .

### 2.2 Mixed Matrix Products

Let  $A$  be an  $n \times n$  matrix over  $R \cup \{+\infty\}$ , and let  $B$  be an  $n \times n$  matrix with entries in  $\{0, 1\}$ . The *mixed right product*  $C$  of  $A$  and  $B$  is defined by

$$C[i, j] = \min\{A[i, k] \mid 1 \leq k \leq n \text{ \& } B[k, j] = 1\} \cup \{+\infty\}$$

If  $C[i, j] \neq +\infty$  then the index  $k$  such that  $C[i, j] = A[i, k]$  (and thus  $B[k, j] = 1$ ) is called a witness for  $C[i, j]$ . Analogously, the *mixed left product*  $C'$  of  $B$  and  $A$  is defined by

$$C'[i, j] = \min\{A[k, j] \mid 1 \leq k \leq n \text{ \& } B[i, k] = 1\} \cup \{+\infty\},$$

and if  $C'[i, j] \neq +\infty$  then the index  $k$  such that  $C'[i, j] = A[k, j]$  is called a witness for  $C'[i, j]$ .

An  $n \times n$  matrix  $W$  such that whenever  $C[i, j] \neq +\infty$  then  $W[i, j]$  is a witnesses for  $C[i, j]$  is called a witness matrix for the right mixed product of  $A$  and  $B$ . Analogously, we define a witness matrix for the left mixed product of  $B$  and  $A$ .

### 2.3 The Reduction

Let  $A_G$  denote the  $n \times n$  adjacency matrix of  $G = (V, E)$ , i.e.,  $A_G[v, u] = 1$  iff  $(v, u) \in E$ .

**Lemma 1.** For an arbitrary  $i \in \{0, 1, \dots, n-2\}$ ,  $D_G^{i+1}$  can be computed on the basis of  $D_G^i$  and the right mixed product of  $D_G^i$  with  $A_G$  or  $D_G^i$  and the left mixed product of  $A_G$  with  $D_G^i$  in time  $O(n^2)$ .

*Proof.* It is sufficient to observe that for any pair  $v, u$  of vertices in  $G$ ,  $D_G^{i+1}[v, u]$  is equal to

$$\min\{D_G^i[v, u], \min\{D_G^i[v, x] + w(u) \mid 1 \leq x \leq n \ \& \ A_G[x, u] = 1\} \cup \{+\infty\}\}$$

Symmetrically,  $D_G^{i+1}[v, u]$  is equal to

$$\min\{D_G^i[v, u], \min\{D_G^i[x, u] + w(v) \mid 1 \leq x \leq n \ \& \ A_G[v, x] = 1\} \cup \{+\infty\}\}$$

□

The following lemma follows the general strategy used to prove Theorem 3.4 in [8].

**Lemma 2.** Let  $G$  be a directed graph  $G$  on  $n$  vertices with nonnegative real vertex weights. Suppose that the right (or left) mixed product of an  $n \times n$  matrix over  $R \cup \{+\infty\}$  with the adjacency matrix  $A_G$  of  $G$  along with the witness matrix can be computed in time  $T_{mix}(n) = \Omega(n^2)$ . The APSP problem for  $G$  can be solved in time  $\tilde{O}(n^{1.5} \sqrt{T_{mix}(n)})$ .

*Proof.* We begin by computing  $D_G^{t-1}$  for some  $t \in [2, \dots, n]$  which will be specified later. By Lemma 1 this computation takes time  $O(tT_{mix}(n))$ .

It remains to determine distances between pairs of vertices where any shortest path consists of at least  $t$  edges. For this purpose, we determine a subset  $B$  of  $V$ , the so called bridging set [26], hitting all the aforementioned long paths. We apply the following fact to  $l = t$  and sets of  $t$  vertices on shortest consisting of exactly  $t-1$  edges, similarly as in [3,8,25,26].

**Fact 1.** Given a collection of  $N$  subsets of  $\{1, \dots, n\}$ , where each subset has size exactly  $l$ , we can find a subset  $B$  of size  $O((n/l) \log n)$  that hits all subsets in the collection in time  $O(Nl)$ .

Since our application of Fact 1 is analogous to those in [3,8,25,26], we solely sketch it referring the reader for details to the aforementioned papers.

Note that for each pair  $v, u$ , of vertices for which any shortest path has at least  $t$  edges there is a pair  $v', u'$  of vertices on a shortest path from  $v$  to  $u$  such that any shortest path from  $v'$  to  $u'$  has exactly  $t-1$  edges. For all such pairs  $v', u'$ , we can find a shortest path on  $t-1$  edges, and thus on  $t$  vertices, by backtracking on the computation of  $D_G^{t-1}$  and using witnesses for the mixed products. In total, we generate  $O(n^2)$  such paths on  $t$  vertices in time  $O(tn^2)$ . The application of Fact 1 also takes time  $O(tn^2)$ .

Next, we run Dijkstra's single-source shortest path algorithm [1] for all vertices in the bridging set  $B$  in the input graph  $G$  and in the graph resulting from reversing the direction of edges in  $G$ . In this way, we determine  $D_G[v, u]$  for all pairs  $(v, u) \in (B \times V) \cup (V \times B)$ .

Now, it is sufficient for all remaining pairs  $(v, u)$  in  $V \times V$  to set

$$D_G[v, u] = \min\{D_G^{t-1}(v, u), \min_{b \in B}\{D_G[v, b] + D_G[b, u] - w(b)\}\}$$

in order to determine the whole  $D_G$ .

The computation of  $D_G^{t-1}$  takes  $O(tT_{mix}(n))$  time which asymptotically is not less than the  $O(tn^2)$  time taken by the construction of the bridging set. The runs of Dijkstra's algorithm and the final computation of  $D_G$  require  $\tilde{O}(\frac{n}{t}n^2)$  time. By setting  $t = \sqrt{\frac{n^3}{T_{mix}(n)}}$ , we obtain the lemma.  $\square$

### 3 Fast Computation of the Mixed Products for Clustered Data

Our algorithm for the right (or, left) mixed product relies on computation of an approximate minimum spanning tree of the columns (or rows, respectively) of the Boolean input matrix in the Hamming metric.

#### 3.1 Approximate Minimum Spanning Tree in High Dimensional Space

For  $c \geq 1$  and a finite set  $S$  of points in a metric space, a *c-approximate minimum spanning tree for S* is a spanning tree in the complete weighted graph on  $S$ , with edge weights equal to the distances between the endpoints, whose total weight is at most  $c$  times the minimum.

In [16] (section 4.3) and [15] (section 3), Indyk and Motwani in particular considered the bichromatic  $\epsilon$ -approximate closest pair problem for  $n$  points in  $R^d$  with integer coordinates in  $O(1)$  under the  $L_p$  metric,  $p \in \{1, 2\}$ . They showed that there is a dynamic data structure for this problem which supports insertions, deletions and queries in time  $O(dn^{1/(1+\epsilon)})$  and requires  $O(dn + n^{1+1/(1+\epsilon)})$ -time preprocessing. In consequence, by a simulation of Kruskal's algorithm they deduced the following fact.

**Fact 2.** *For  $\epsilon > 0$ , a  $1 + \epsilon$ -approximate minimum spanning tree for a set of  $n$  points in  $R^d$  with integer coordinates in  $O(1)$  under the  $L_1$  or  $L_2$  metric can be computed by a Monte Carlo algorithm in time  $O(dn^{1+1/(1+\epsilon)})$ .*

In [17] Indyk, Schmidt and Thorup reported even slightly more efficient (by a poly-log factor) reduction of the problem of finding a  $1 + \epsilon$ -approximate minimum spanning tree to the bichromatic  $\epsilon$ -approximate closest pair problem via an easy simulation of Prim's algorithm.

Note that the  $L_1$  metric for points in  $R^n$  with 0, 1-coordinates coincides with the  $n$ -dimensional Hamming metric. Hence, Fact 2 immediately yields the following corollary.

**Corollary 1.** *For  $\epsilon > 0$ , a  $1 + \epsilon$ -approximate minimum spanning tree for a set of  $n$  0-1 strings of length  $n$  under the Hamming metric can be computed by a Monte Carlo algorithm in time  $O(n^{2+1/(1+\epsilon)})$ .*

### 3.2 The Algorithm for Mixed Matrix Product

The idea of our combinatorial algorithm for the right mixed product  $C$  of  $A$  with  $B$  and its witness matrix is a generalization of that from [5]. Let  $P(r, v)$  denote a priority queue (implemented as a heap) on the entries  $A[r, k]$  such that  $B[k, v] = 1$  ordered by their values in nondecreasing order.

First, we compute an approximate minimum spanning tree of the columns of  $B$  in the Hamming metric. Then, we fix a traversal of the tree. Next, for each row  $r$  of  $A$ , we traverse the tree, construct  $P(r, start)$  where  $start$  is the first column of  $B$  in the tree traversal and then maintain  $P(r, v)$  for the currently traversed  $v$  by updating  $P(r, u)$  where  $u$  is the predecessor of  $v$  in the traversal. A minimum element in  $P(r, v)$  yields a witness for  $C[r, v]$ . The cost of the updates in a single traversal of the tree is proportional to the cost of the tree modulo a logarithmic factor.

#### Algorithm 1

**Input:**  $n \times n$  matrix  $A$  over  $R \cup \{+\infty\}$  and an  $n \times n$  Boolean matrix  $B$ ;

**Output:** A witness matrix  $W$  for the right mixed product  $C$  of  $A$  and  $B$ .

**Comment:**  $P(r, v)$  stands for a priority queue on the entries  $A[r, k]$  s.t.  $B[k, v] = 1$  ordered by their values in nondecreasing order.

1. Compute an  $O(\log n)$ -approximate minimum spanning tree  $T_B$  of the columns of  $B$  in the Hamming metric;
2. Fix a traversal of the tree  $T_B$  linear in its size;
3. Set  $start$  to the first node of the traversal;
4. For each pair of consecutive neighboring columns  $v, u$  in the traversal, precompute the set  $D_{v,u}$  of positions where 1s occur in  $v$  but not in  $u$  and the set  $D_{u,v}$  of positions where 1s occur in  $u$  but not in  $v$ ;
5. For each row  $r$  of  $A$  do
  - Construct the priority queue  $P(r, start)$  and if  $P(r, start) \neq \emptyset$  set  $W[r, start]$  to the index  $k$  where  $A[r, k]$  is the minimum element in  $P(r, start)$ ;
  - Traverse the tree  $T_B$  and for each node  $v$  different from  $start$  compute the priority queue  $P(r, v)$  from the priority queue  $P(r, u)$ , where  $u$  is the predecessor of  $v$  in the traversal, by utilizing  $D_{v,u}$  and  $D_{u,v}$ . If  $P(r, v) \neq \emptyset$  set  $W[r, v]$  to the index  $k$  where  $A[r, k]$  is the minimum element in  $P(r, v)$ .

**Lemma 3.** *Algorithm 1 is correct, i.e., it outputs the witnesses matrix for the right mixed product of matrices  $A$  and  $B$ .*

For an  $n \times n$  Boolean matrix  $C$ , let  $K_C$  be the complete weighted graph on the rows of  $C$  where the weight of an edge between two rows is equal to their Hamming distance. Next, let  $MWT(C)$  be the weight of a minimum weight spanning tree of  $K_C$ .

**Lemma 4.** *Algorithm 1 can be implemented in time  $\tilde{O}(n(n + MWT(B^t))) + t(n)$ , where  $t(n)$  is the time taken by the construction of the  $O(\log n)$ -approximate minimum weight spanning tree in step 1.*

*Proof.* Step 1 can be implemented in time  $t(n)$  while steps 2,3 take time  $O(n)$ . Step 4 takes  $O(n^2)$  time. The block in Step 5 is iterated  $n$  times.

The first step in the block, i.e., the construction of  $P(r, \text{start})$  takes  $O(n \log n)$  time. The update of  $P(r, u)$  to  $P(r, v)$  takes  $O(\log n(|D_{v,u}| + |D_{u,v}|))$  time. Note that  $|D_{v,u}| + |D_{u,v}|$  is precisely the Hamming distance between the columns  $v$  and  $u$ . It follows by the  $O(\log n)$  approximation factor of  $T_B$  that the total time taken by these updates is  $O(MWT(B^t) \log^2 n)$ .

We conclude that Step 5 can be implemented in time  $\tilde{O}(nMWT(B^t))$ .  $\square$

**Theorem 1.** *The right mixed product of two  $n \times n$  matrices  $A$  over  $R \cup \{+\infty\}$  and  $B$  over  $\{0, 1\}$  can be computed by a combinatorial randomized algorithm in time  $\tilde{O}(n(n+MWT(B^t)))$ . Analogously, the left mixed product of  $B$  and  $A$  can be computed by a combinatorial randomized algorithm in time  $\tilde{O}(n(n+MWT(B)))$ .*

*Proof.* By Corollary 1, an  $\Theta(\log n)$ -approximate minimum spanning tree can be constructed by a Monte Carlo algorithm in time  $\tilde{O}(n^2)$  (observe that  $n^{1/f} = O(1)$  if  $f = \Omega(\log n)$ ). Hence, by Lemmata 3, 4, we obtain the theorem for the right mixed product. The upper bound on the time required to compute the left mixed product follows symmetrically.  $\square$

## 4 Main results

Lemma 2 combined with Theorem 1 yield our main result.

**Theorem 2.** *Let  $G$  a directed graph  $G$  on  $n$  vertices with nonnegative real vertex weights. The all-pairs shortest path problem for  $G$  can be solved by a combinatorial randomized algorithm in time  $\tilde{O}(n^2 \sqrt{n + \min\{MWT(A_G), MWT(A_G^t)\}})$ .*

By setting vertex weights, say, to zero, we obtain immediately the following corollary.

**Corollary 2.** *The transitive closure of a directed graph  $G$  on  $n$  vertices can be computed by a combinatorial randomized algorithm in time  $\tilde{O}(n^2 \sqrt{n + \min\{MWT(A_G), MWT(A_G^t)\}})$ .*

Equivalently, we can formulate Corollary 2 as follows.

**Corollary 3.** *The transitive closure of an  $n \times n$  Boolean matrix  $B$  (over the Boolean semi-ring) can be computed by a combinatorial randomized algorithm in time  $\tilde{O}(n^2 \sqrt{n + \min\{MWT(B), MWT(B^t)\}})$ .*

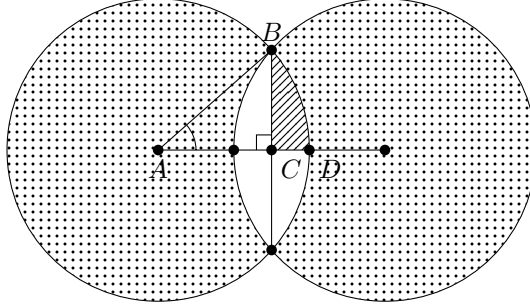
## 5 APSP in vertex-weighted uniform disk graphs of bounded density

In this section, we consider uniform disk graphs that are induced by a set  $P$  of  $n$  points in a unit square in the plane that are  $b(n)$ -dense, where  $b : N \rightarrow N$ .



Formally, we say that  $P$  is  $b(n)$ -dense iff each cell of the regular  $\sqrt{n} \times \sqrt{n}$  grid within the unit square contains at most  $b(n)$  points. The vertices of such an induced disk graph are the points in  $P$ , and two vertices are adjacent in the graph iff their Euclidean distance is at most  $r$ , where  $r$  is a positive constant not exceeding 1. We shall term the aforementioned graphs as *uniform disk graphs induced by  $b(n)$ -dense point sets*.

**Lemma 5.** *Given two intersecting disks on the plane of the same radius  $r$  with the distance  $d$  between centers, the area of the symmetric difference is  $O(rd)$ .*



*Proof.*  $AC = \frac{d}{2}$ ,  $AB = r$ .

The area of the triangle  $ABC$  is

$$Area_{ABC} = \frac{1}{2} AC BC = \frac{1}{2} \frac{d}{2} \sqrt{r^2 - \frac{d^2}{4}} = \frac{1}{8} d \sqrt{4r^2 - d^2}.$$

The area of the circular sector  $ABD$  is

$$Area_{ABD} = \frac{1}{2} r^2 \angle BAC = \frac{1}{2} r^2 \arccos\left(\frac{d}{2r}\right).$$

The area of  $BCD$  is  $Area_{BCD} = Area_{ABD} - Area_{ABC}$ .

The area of the symmetric difference

$$Area = 2(\pi r^2 - 4Area_{BCD}) = 2\pi r^2 - 4r^2 \arccos\left(\frac{d}{2r}\right) + d\sqrt{4r^2 - d^2}.$$

Finally, by using Taylor series expansion

$$\begin{aligned} 4r^2 \arccos\left(\frac{d}{2r}\right) &= 4r^2 \left( \frac{\pi}{2} - \frac{d}{2r} + O\left(\left(\frac{d}{2r}\right)^2\right) \right) = \\ &= 2\pi r^2 - 2dr + O(d^2) = 2\pi r^2 - 2dr + O(rd) \end{aligned}$$

and  $\sqrt{4r^2 - d^2} \leq 2r$  we get  $Area = O(rd)$ . □

**Lemma 6.** *Let  $G$  be a uniform disk graph induced by a  $b(n)$ -dense point set. For each edge  $(v, u)$  of  $G$ , the number of vertices in  $G$  that are a neighbor of exactly one of the vertices  $v, u$ , i.e., the Hamming distance between the two rows in the adjacency matrix of  $G$  corresponding to  $v$  and  $u$ , respectively, is  $O(r \times b(n)(\text{dist}(v, u) \times n + \sqrt{n}))$ .*

*Proof.* The number of vertices of  $G$  that are a neighbor of exactly one of the vertices  $v$  and  $u$  is at most the minimum number of cells of the regular  $\sqrt{n} \times \sqrt{n}$  grid within the unit square that cover the symmetric difference  $S(v, u)$  between the disks centered at  $v$  and  $u$ , respectively, multiplied by  $b(n)$ . The aforementioned number of cells is easily seen to be at most the area  $A(v, u)$  of  $S(v, u)$  divided by the area of the grid cell, i.e.,  $A(v, u) \times n$ , plus the number of cells of the grid intersected by the perimeter of  $S(v, u)$ , i.e.,  $O(r\sqrt{n})$ . By Lemma 5, we have  $A(v, u) = O(\text{dist}(v, u) \times r)$ . Hence, the aforementioned number of cells is  $O(r(\text{dist}(v, u) \times n + \sqrt{n}))$ .  $\square$

The following lemma is a folklore (e.g., it follows directly from the upper bound on the length of closed path through a set of points in a  $d$ -dimensional cube given in Lemma 2 in [18]).

**Lemma 7.** *The minimum Euclidean spanning tree of any set of  $n$  points in a unit square in the plane has total length  $O(\sqrt{n})$ .*

Combining Lemmata 6, 7, we obtain the following one.

**Lemma 8.** *For a uniform disk graph  $G$  induced by a  $b(n)$ -dense  $n$ -point set, a spanning tree of the rows (or, columns) of the adjacency matrix of  $G$  in the Hamming metric having cost  $O(rn^{3/2})$  can be found in time  $O(n^2)$ .*

*Proof.* Construct a minimum Euclidean spanning tree of the  $n$  points forming the vertex set of  $G$ . It takes time  $O(n \log n)$  and the resulting tree  $T$  has total length  $O(\sqrt{n})$  by Lemma 7. Form a spanning tree  $U$  of the rows (or, columns) of the adjacency matrix of  $G$  by connecting by edge the rows corresponding to  $v$  and  $u$  iff  $(v, u) \in T$ . By Lemma 6 and the  $O(\sqrt{n})$  length of  $T$ , the total cost of  $U$  is  $O(rn^{3/2}b(n))$ .  $\square$

By plugging Lemma 8 into Theorem 2, we obtain our main result in this section.

**Theorem 3.** *Let  $G$  be a uniform disk graph, with nonnegative real vertex weights, induced by a  $b(n)$ -dense  $n$ -point set. The all-pairs shortest path problem for  $G$  can be solved by a combinatorial algorithm in time  $\tilde{O}(\sqrt{r}n^{2.75}\sqrt{b(n)})$ .*

In the application of the method of Theorem 2 yielding Theorem 3, we can use the deterministic algorithm of Lemma 8 to find a spanning tree of the rows or columns of the adjacency matrix of  $G$  instead of the randomized approximation algorithm from Fact 2.

By straightforward calculations, our upper time-bound for APSP in vertex-weighted uniform disk graphs induced by  $O(1)$ -dense point sets subsumes that for APSP in sparse graphs based on Dijkstra's single-source shortest-path algorithm, running in time  $\tilde{O}(nm)$ , where  $m$  is the number of edges, for  $r \gg n^{-1/6}$ .

Finally, we can also easily extend Theorem 3 to include uniform ball graphs in a  $d$ -dimensional Euclidean space. In the extension, the term  $\sqrt{r}$  in the upper time-bound generalizes to  $\sqrt{r^{d-1}}$ .

## 6 Final Remarks

We can easily extend our main result to include solving the APSP problem for vertex and edge weighted directed graphs in which the number of different edge weights is bounded, say by  $q$ . This can be simply achieved by decomposing the adjacency matrix  $A_G$  into the union of up to  $q$  matrices  $A_1, A_2, \dots, A_l$  in one-to-one correspondence with the distinct edge weights and consequently replacing each mixed product with  $l$  such products in Lemmata 1, 2. In the final upper bound,  $MWT(A_G)$  and  $MWT(A_G^t)$  are replaced by  $\sum_{i=1}^l MWT(A_i)$  and  $\sum_{i=1}^l MWT(A_i^t)$ , respectively.

It is an interesting problem to determine if there are other natural graph classes where  $MWT(A_G)$  or  $MWT(A_G^t)$  are substantially subquadratic in the number of vertices.

It follows from the existence of the so called Hadamard matrices [7] that there is an infinite sequence of graphs with  $n_i \times n_i$  adjacency matrices  $A_i$  such that  $\min\{MWT(A_i), MWT(A_i^t)\} = \Omega((n_i)^2)$  holds.

## References

1. A.V. Aho, J.E. Hopcroft and J.D. Ullman. The Design and Analysis of Computer Algorithms (Addison-Wesley, Reading, Massachusetts, 1974).
2. N. Alon and M. Naor. Derandomization, Witnesses for Boolean Matrix Multiplication and Construction of Perfect hash functions. *Algorithmica* 16, pp. 434-449, 1996.
3. N. Alon, Z. Galil and O. Margalit. On the exponent of all pairs shortest path problem. *J. Comput. System Sci.*, 54 (1997), pp. 25-51.
4. N. Bansal and R. Williams. Regularity Lemmas and Combinatorial Algorithms. *Proc. of 50th IEEE Symposium on Foundations on Computer Science*, Atlanta 2009.
5. A. Björklund and A. Lingas. Fast Boolean matrix multiplication for highly clustered data. *Proc. 7th International Workshop on Algorithms and Data Structures (WADS 2001)*, Lecture Notes in Computer Science, Springer Verlag.
6. A. Borodin, R. Ostrovsky and Y. Rabani. Subquadratic Approximation Algorithms For Clustering Problems in High Dimensional Spaces. *Proceedings of the 31st ACM Symposium on Theory of Computing*, 1999.
7. P.J. Cameron. *Combinatorics*. Cambridge University Press 1994.
8. T.M. Chan. More algorithms for all-pairs shortest paths in weighted graphs. *SIAM J. Comput.* Vol. 39, No. 5, pp. 2075-2089 (preliminary version in *proc. STOC 2007*, pp. 590-598).

9. T.M. Chan. All-pairs shortest paths with real weights in  $O(n^3/\log n)$  time. *Algorithmica* 41 (2008), pp. 330-337.
10. D. Coppersmith and S. Winograd. Matrix Multiplication via Arithmetic Progressions. *J. of Symbolic Computation* 9 (1990), pp. 251-280.
11. M. Fürer and P. Kasiviswanathan. Approximate Distance Queries in Disk Graphs. *Proc. Workshop on Approximation and Online Algorithms (WAOA 2006)*, Lecture Notes in Computer Science 4368, pp. 174-187, 2006.
12. Z. Galil and O. Margalit. Witnesses for Boolean Matrix Multiplication and Shortest Paths. *Journal of Complexity*, pp. 417-426, 1993.
13. L. Gasieniec and A. Lingas, An improved bound on Boolean matrix multiplication for highly clustered data. *Proc. 9th International Workshop on Algorithms and Data Structures (WADS 2003)*, Lecture Notes in Computer Science 2748, pp. 329-339, Springer Verlag.
14. X. Huang and V.Y. Pan. Fast rectangular matrix multiplications and applications. *Journal of Complexity* 14(2), pp. 257-299 (1998).
15. P. Indyk. High-dimensional computational geometry. PhD dissertation, Stanford University, 2000.
16. P. Indyk and R. Motwani. Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality. *Proceedings of the 30th ACM Symposium on Theory of Computing*, 1998.
17. P. Indyk, S.E. Schmidt, and M. Thorup. On reducing approximate mst to closest pair problems in high dimensions. *Manuscript*, 1999.
18. R. M. Karp and J. M. Steele. Probabilistic analysis of heuristics. Chapter 6 in *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*, pp. 181-205 (edited by E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys). John Wiley & Sons Ltd., 1985.
19. E. Kushilevitz, R. Ostrovsky, and Y. Rabani. Efficient search for approximate nearest neighbor in high dimensional spaces. *SIAM J. Comput.* 30, 2, pp. 457-474. (Preliminary version in *Proc. 30th STOC*, 1989.)
20. A. Lingas. A geometric approach to Boolean matrix multiplication. *Proc. 13th International Symposium on Algorithms and Computation (ISAAC 2002)*, Lecture Notes in Computer Science 2518, Springer Verlag, pp. 501-510.
21. J. I. Munro. Efficient determination of the transitive closure of a directed graph. *Information Processing Letters* 1(2), pp. 56-58, 1971.
22. W. Rytter. Fast recognition of pushdown automaton and context-free languages. *Information and Control* 67(1-3), 12-22, (1985).
23. R. Seidel. On the All-Pairs-Shortest-Path Problem. *Proc. 24th ACM STOC*, 1992, pp. 745-749.
24. V. Vassilevska Williams and R. Williams. Subcubic Equivalences Between Path, Matrix, and Triangle Problems. In: *Proceedings 51st Annual IEEE Symposium on Foundations of Computer Science (FOCS 2010)*.
25. R. Yuster. Efficient algorithms on sets of permutations, dominance, and real-weighted APSP. *Proc. of the 20th ACM-SIAM Symposium on Discrete Algorithms*, 2009, pp. 950-957.
26. U. Zwick. All pairs shortest paths using bridging rectangular matrix multiplication. *Journal of the ACM*, 49(3), pp. 289-317, 2002.
27. U. Zwick. Exact and Approximate Distances in Graphs - A survey. *Proc. ESA* 2001, pp. 33-48.